Control and Management of a Connected Car Using YANG/RESTCONF and Cloud Computing

Ricard Vilalta, Selva Via, Fermín Mira, Luis Sanabria, Ricardo Martínez, Ramon Casellas,
Raul Muñoz, and Jesus Alonso-Zarate
Centre Tecnologic de Telecomunicacions de Catalunya (CTTC/CERCA), Spain
ricard.vilalta@cttc.es

***Abstract*—This paper describes the implementation of an innovative proof-of-concept (PoC) for a connected car, modeled with YANG, which can be remotely controlled using SDN/NFV technologies. In particular, the remote control of the car is based on a service application running on a remote data center. The demonstration is performed using a RESTCONF server installed in a Raspberry Pi aboard of a small car. This server is responsible for the sensors and actuators of the car and allows for its remote control from a user terminal (e.g., a smartphone, tablet, or laptop) and through the cloud, running a control application as a service.**

***Index Terms*—Connected Car, 5G, SDN, NFV, YANG**

I. Introduction

It is foreseen that in the near future all vehicles will be connected. Vehicles may be connected to the Internet, to their peers (V2V), pedestrians (V2P), the road infrastructure (V2I), or the communication network (V2N). The communication between two entities where at least one of them is a vehicle is typically referred to as Vehicle-to-Anything (V2X) communications. This type of communications will improve road safety, increase traffic efficiency, and enhance user comfort. New services will be created on top of this connectivity, thus extending the automotive ecosystem to new players entering into this business domain [1]. However, many challenges are still unsolved. One of the key open questions to be solved is how communication networks should be tailored for V2X communications. It is clear that networks of the future should deal with at least 3 of the key challenges posed by the Internet of Things (IoT): 1) need for interoperability, 2) coexistence of a wide variety of sometimes opposite requirements defined by different applications, and 3) need to support high scalability [2]. Indeed, 5G technologies are being designed to cope with all these requirements.

In view of this, and understanding the connected car as a complex cyber-physical system (CPS), many of the communication techniques which have been designed for the Internet of Things (IoT) can be applied and adapted (i.e., redesigned) to those networks which will handle the connected vehicle. This includes, among others, optimized wireless communication protocols, data formatting protocols, cloud computing, Software-Defined Networking (SDN), Network Function Virtualization (NFV), etc. Cloud computing can help processing the data gathered by billions of smart things interacting with each other. Further, the SDN paradigm enables a global orchestration of all network resources including, for example, the management of distributed clouds and the coexistence of heterogeneous networks combining different types of communication technologies. In its turn, NFV has introduced a novel paradigm where services can be deployed on demand in order to fulfill the end user needs. These three techniques are intertwined: in new communication networks, services are deployed over a cloud computing infrastructure, where the necessary connectivity is provided by an SDN controller. The authors have previously proposed in [3], the usage of a service orchestrator for IoT applications.

Under this context, the SDN orchestrator must carry out the following 3 key functions: i) facilitate the transport of the huge amount of data generated at the terminals, sensors, machines, nodes, etc., to any distributed computing node, edge, or core data center; ii) allocate computing and storage resources in distributed data centers, and iii) process the collected data (Big Data) to make proper decisions, leading to the concept of cognition [3].

Beyond connectivity, the ultimate key element here is the data, from which real value can be obtained. In the end, connectivity is just the means to gather and obtain the data. So, when it comes to processing and operating the data, formatting this data becomes a key design decision. Indeed, the adoption of a common, flexible, and powerful data and information modeling language to define all sensors, actuators, gateway facilities and services is a first important step towards the standardisation of IoT frameworks across multiple vendors beyond the existing ones. The automotive sector is not an exception to this.

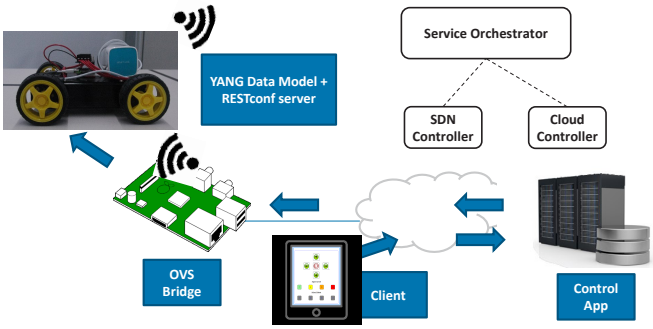


Fig. 1. Connected Car demonstration architecture

Among other options, over the last years, YANG has been steadily growing in the IT and networking communities as a data modeling language suitable for the IoT [4]. For such purpose, YANG data models need to be complemented with NETCONF/RESTCONF protocols [5]. These protocols enable the control and management of YANG data models.

Motivated by all this context, this paper describes the first proof-of-concept of a remotely-controlled car through and SDN/NFV-enabled communications network where YANG and NETCONF/RESTCONF have been used to model, manage, and control the data associated to the car. The remote control is based on an application running on a cloud infrastructure which can be accessed from any user terminal, e.g. smartphone, tablet, laptop, etc. In particular, the demonstration is performed using a small toy car equipped with an on-board Raspberry Pi which runs a RESTCONF server responsible for the interaction with the sensors and actuators of the car.

The remainder of the paper is organized as follows. The proposed architecture is described in Section II. Section III is devoted to describe the demo set-up and discuss the key demonstrated results. Finally, Section IV provides conclusions.

II. Proposed Architecture

Figure 1 shows the proposed architecture for the control and management of the connected car.

The Service Orchestrator acts as a generalized NFV Manager and Orchestrator (NFV MANO), and is responsible for: 1) Triggering the necessary flows in the SDN controller in order to interconnect the OpenVSwitch (OVS) bridge, 2) requesting the necessary resources and deploying the requested services on top of them, and 3) deploying services, which are independent from each other, using the Cloud Controller. In our particular case, the control application is a service that the Service Ochestrator deploys. The Cloud Controller (e.g., based on OpenStack) acts as a Virtualized Infrastructure Manager (VIM) in the proposed NFV architecture. The VIM is responsible for the creation/migration/deletion of virtual machine (VM) instances (computing service), disk images storage (image service), and the management of the VM network interfaces (networking service). The computing service (e.g., Nova in OpenStack) is responsible for the management of the VM into the compute hosts. A compute service agent is running in each host and controls the computing hypervisor (e.g., KVM) responsible for the creation/deletion of the VMs. The Control Application is stored as an image, which is deployed on demand, when requested by the Service Orchestrator. An SDN controller (e.g., ONOS) is the responsible for the control of the network resources (such as the OVS bridge). The SDN controller translates high level connectivity intent requests into OpenFlow (OF) protocol commands, in order to configure the necessary underlying network resources for the establishment of the requested connection.

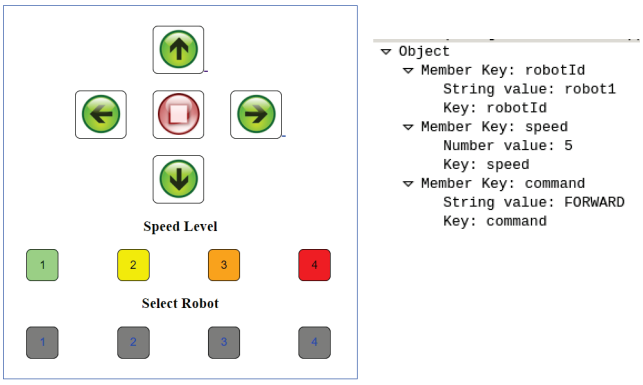


Fig. 2. Example of Connected Car UI and JSON command

The connected car offers a RESTCONF server which can be accessed through a control application being accessed from a user terminal, i.e. the client. The graphical user interface (GUI) of this control application is shown in Figure 2.

The connected car data model is described in YANG modeling language. RESTCONF is used as transport protocol which uses JavaScript Object Notation (JSON) encoding for data transmission. RESTCONF is an HTTP-based protocol for configuring data defined in YANG. It uses HTTP methods (i.e., POST, PUT, PATCH, and DELETE) to provide Create Read Update and Delete (CRUD) operations on a conceptual datastore containing YANG-defined data. RESTCONF combines the simplicity of HTTP with the predictability and automation potential of a schema-driven API.

In the implementation described in this paper, the connected car data model is composed of three main parameters (shown in Fig. 2): 1) robotId, 2) speed, and 3) command. The allowed commands are: FORWARD, STOP, BACKWARD, LEFT, and RIGHT, indicating the direction of motion of the car.

III. Demo set-up and results

The demo toy-car is composed of a metal chassis and 4 wheels each driven by 4 DC motors powered with dedicated 5 AA batteries in series, providing 7.5V. The car is equipped with a Raspberry Pi Zero W (RasPiZW) [6]. RasPiZW is a basic computer consisting of a single-core 1GHz ARM11 processor (Broadcom SoC BCM2835), with 512MB of RAM. This version offers 802.11n wireless LAN and Bluetooth 4.1 and BLE wireless connectivity with a Cypress CYW43438 chip and a 2.4GHz PCB antenna. It needs constant 5V power supply via a micro USB connector. Power consumption in Idle state is about 120mA. The RasPiZW in the car is supplied with a battery pack. The operative system, based on Debian Jessie, is Raspbian Jessie Lite. This minimal image suits the functionality of the car, where no graphical interface is needed, allowing more efficient operation of the RasPiZW. To control the motor, a commercial add-on for raspberry Pi has been used, MotoZero. MotoZero is based on two Texas Instruments motor driver ICs, LD293D, to allow control of each wheel independently.

The OVS bridge is implemented using a Raspberry Pi 3, including various Ethernet to USB adapters. OpenVSwitch is setup in order to control the data plane Ethernet connections.

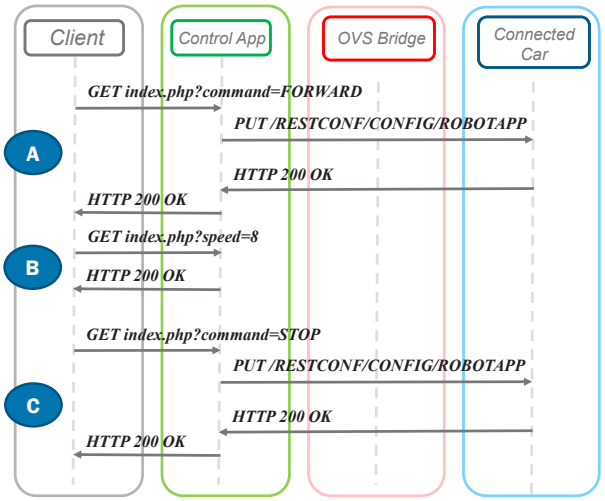


Fig. 3. Proposed Message exchange workflow

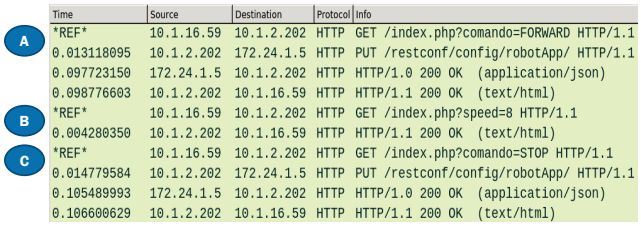
The Cloud Controller has been simplified in the demonstration, being a server running an HTTP Apache server with PHP extensions. In further demonstrations a full OpenStack Cloud Computing service will be demonstrated.

The RESTCONF server has been autogenerated with OpenSourceSDN.Org project EAGLE YANG to code tools [7]. These tools allow for the rapid prototyping of RESTCONF servers using the feed from YANG data models.

Figure 3 presents the suggested workflow for the control and management of the connected car using YANG/RESTCONF. The Control Application is a PHP application, which shows the Control UI, and translates the pressed commands into the necessary HTTP RESTCONF commands for the control of the connected car. At least three scenarios are envisioned:

- Scenario A: Move the connected car. A Forward command is sent from the client to the Control Application. The Control Application issues the RESTCONF HTTP PUT operation, including the received command, the selected robotId, as well as the previously configured speed for the motors. The OVS bridge forwards the command, and the Connected Car RESTCONF server processes the command and activates the necessary motors. Once activated, the HTTP 200 OK command is answered to the Control Application which, in its turn, notifies the client.
- Scenario B: Speed selection. In this scenario, the client selects the speed for a certain robotId. This information is received by the Control Application and it is stored in a local MySQL database. No interaction with the Connected Car is performed in this case.
- Scenario C: Stop the connected car. Finally, the client requests the stop command to the Control Application. As in Scenario A, the Control Application issues the RESTCONF HTTP PUT operation, including the received command and the selected robotId.

Figure 4 shows the captured Wireshark traces for the presented use cases. These Wireshark traces demonstrate the



Time	Source	Destination	Protocol	Info
REF	10.1.16.59	10.1.2.202	HTTP	GET /index.php?comando=FORWARD HTTP/1.1
0.013118095	10.1.2.202	172.24.1.5	HTTP	PUT /restconf/config/robotApp/ HTTP/1.1
0.097723150	172.24.1.5	10.1.2.202	HTTP	HTTP/1.0 200 OK (application/json)
0.098776603	10.1.2.202	10.1.16.59	HTTP	HTTP/1.1 200 OK (text/html)
REF	10.1.16.59	10.1.2.202	HTTP	GET /index.php?speed=8 HTTP/1.1
0.004280350	10.1.2.202	10.1.16.59	HTTP	HTTP/1.1 200 OK (text/html)
REF	10.1.16.59	10.1.2.202	HTTP	GET /index.php?comando=STOP HTTP/1.1
0.014779584	10.1.2.202	172.24.1.5	HTTP	PUT /restconf/config/robotApp/ HTTP/1.1
0.105489993	172.24.1.5	10.1.2.202	HTTP	HTTP/1.0 200 OK (application/json)
0.106600629	10.1.2.202	10.1.16.59	HTTP	HTTP/1.1 200 OK (text/html)

Fig. 4. Captured message exchange with wireshark

feasibility of the proposed approach, based on using YANG-based services for Connected Car applications. Latency results have been obtained through a WiFi network. It is foreseen that latency will be significantly reduced when new radio access networks, such as 5G, will be use instead of WiFi.

IV. Conclusion

In this paper, we have described the first implementation of a proof-of-concept for a remotely-controlled car using YANG modeling for the data associated to the car, and operating an SDN/NFV network with dynamic service provisioning. A user can interact with the car through an application with a user interface which is served by a network orchestrator and can be operated from any handheld or desktop device. This proof-of-concept has demonstrated: first, the feasibility of the proposed approach of using YANG modeling language for the description of IoT services, and second, the suitability of unifying the NFV paradigm together with IoT services.

Acknowledgment

Part of this work has been performed in the framework of the H2020 project 5GCAR co-funded by the EU. Authors would like to acknowledge the contributions of their colleagues from 5GCAR although the views expressed are those of the authors and do not necessarily represent the views of the 5GCAR project. This research also has been partly funded by Spanish MINECO projects DESTELLO (TEC2015-69256-R) and CELLFIVE (TEC2014-60130-P).

References

[1] B. Martinez de Aragon, J. Alonso-Zarate, A. Laya, "How Connectivity is Transforming the Automotive Ecosystem," *Wiley Internet Letters*, in press 2017.

[2] NGMN Alliance, "NGMN 5G White Paper," NGMN Alliance, February 2015, Tech. Rep.

[3] R. Vilalta, I. Popescu, A. Mayoral, X. Cao, R. Casellas, N. Yoshikane, R. Martínez, T. Tsuritani, I. Morita, and R. Muñoz, "End-to-end sdn/nfv orchestration of video analytics using edge and cloud computing over programmable optical networks," in *Optical Fiber Communications Conference and Exhibition (OFC), 2017*. IEEE, 2017, pp. 1–3.

[4] M. Yannuzzi, F. van Lingen, A. Jain, O. L. Parellada, M. M. Flores, D. Carrera, J. L. Pérez, D. Montero, P. Chacin, A. Corsaro *et al.*, "A new era for cities with fog computing," *IEEE Internet Computing*, vol. 21, no. 2, pp. 54–67, 2017.

[5] A. Bierman, M. Bjorklund, and K. Watsen, "Restconf protocol," 2017.

[6] E. Upton and G. Halfacree, *Raspberry Pi user guide*. John Wiley & Sons, 2014.

[7] OpenSourceSDN.org, "EAGLE Project," https://github.com/OpenNetworkingFoundation/EAGLE-Open-Model-Profile-and-Tools/, 2017.